LÓGICA DE PROGRAMAÇÃO

Centro Universitário UniSenac – Campus Pelotas

Escola de Tecnologia da Informação

Prof. Edécio Fernando Iepsen

REPETIÇÕES

- As repetições permitem que um comando ou um conjunto de comandos sejam executados várias vezes.
- Estas estruturas são também conhecidas como *loops* ou laços de repetição.
- As repetições precisam de um ponto de interrupção – que pode ser indicada pelo usuário ou a partir de uma configuração do sistema.

EXEMPLOS

- Calcular e exibir as parcelas de um financiamento de uma casa ou um veículo.
- Solicitar a senha de um usuário, enquanto ele não acertar ou atingir um número x de vezes.
- Montar o *layout* de uma página de comércio eletrônico, a partir da exibição dos produtos disponíveis na loja.

TIPOS DE REPETIÇÕES

- Repetições com variável de controle: laços for
- Repetições com teste no início: laços while
- Repetições com teste no final: laços do... while.

REPETIÇÕES COM VARIÁVEL DE CONTROLE

A sintaxe do comando `for` é composta de três instruções, que definem: a) o valor inicial da variável de controle; b) a condição que determina se a repetição deve ou não continuar; c) o incremento ou decremento da variável de controle. A Figura 5.1 destaca a sintaxe de um comando `for` com valores de exemplo atribuídos a uma variável de controle `i`.

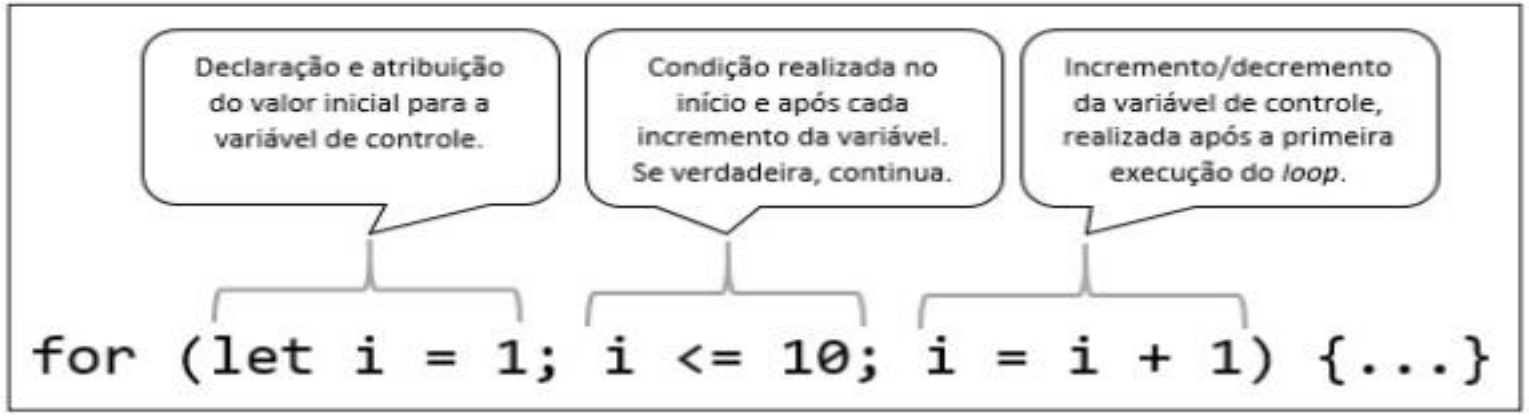

Figura 5.1 – Sintaxe do comando for com o significado de cada parte.

Entre as chaves `{}` devem ser inseridos os comandos que serão executados repetidas vezes. O incremento `i = i + 1` pode ser abreviado por `i++`. A repetição é controlada por uma variável, que, no exemplo da Figura 5.1, inicia em 1 e aumenta até 10.

Poderíamos comparar o loop executado pelo comando `for` a um maratonista percorrendo as voltas de uma pista de atletismo. Na execução do `for`, quando se realiza uma volta completa, o valor da variável é incrementado e essa sequência de passos é repetida até o final. O maratonista realiza um processo semelhante, pois, ao passar pelo ponto de partida, tem o seu número de voltas incrementado em 1. Os loops do `for` contêm, na sua variável `i`, o valor indicativo da volta sendo realizada. A Figura 5.2 ilustra o funcionamento de um loop `for`, que contém inclusive uma condição que avalia o valor da variável `i`, retornando verdadeiro nas voltas de número par.

```
for (let i = 1; i <= 10; i++){
  comando1
  comando2
  comando3
  if (i % 2 == 0) {
    comando4
  }
}
```

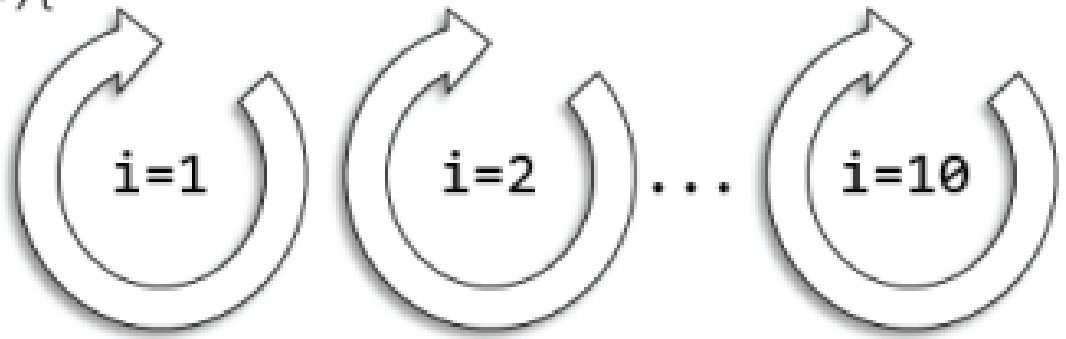

Figura 5.2 – A cada loop, a variável de controle "i" é incrementada em um.

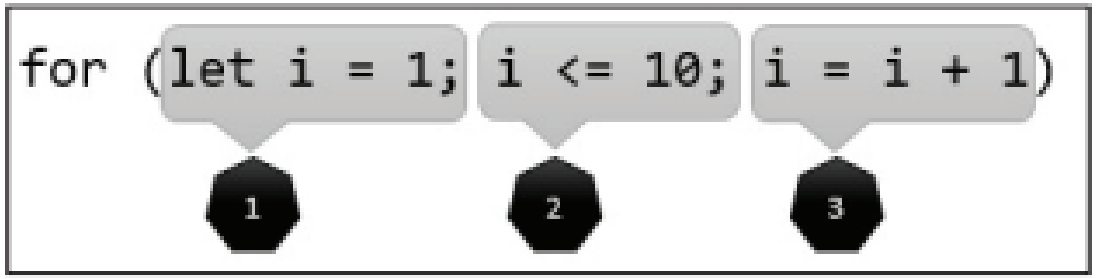

Figura 5.3 – Ao entender a sequência de execução das partes que compõem o comando for, você poderá montar diversas variações de valores para a variável de controle.

A sequência de execução das instruções é a seguinte: 1 e 2 (executa), 3 e 2 (executa), 3 e 2 (executa), ... (segue 3 e 2 até a condição ficar falsa).

REPETIÇÕES COM TESTE NO INÍCIO

Um laço de repetição também pode ser criado com o comando `while`, que realiza um teste condicional logo no seu início, para verificar se os comandos do laço serão ou não executados. A tradução da palavra `while`, que em português significa enquanto, define bem o seu funcionamento: "enquanto a condição for verdadeira, execute". A sintaxe do comando `while` é a seguinte:

```
while (condição) {
  comandos
}
```

As estruturas de repetição com teste no início, representadas pelo comando `while`, são utilizadas principalmente em programas que manipulam arquivos, para repetir a leitura de uma linha enquanto não atingir o final do arquivo.

REPETIÇÕES COM TESTE NO FINAL

Outra forma de criar laços de repetição em um programa é com a utilização do comando `do.. while`, cuja sintaxe é representada a seguir:

```
do {
  comandos
} while (condição)
```

Uma sútil, porém importante, diferença entre as estruturas de repetição `while` e `do.. while` é a seguinte: com o comando `while`, a condição é verificada no início; enquanto, com o comando `do.. while`, a condição é verificada no final. Ou seja, com o `do.. while`, fica garantido que uma vez, no mínimo, os comandos que pertencem ao laço serão executados.

Geralmente optamos por utilizar os laços `while` e `do.. while` quando não soubermos previamente quantas vezes a repetição vai ocorrer. Pense em algo como o processo do recebimento de contas realizado por um terminal de caixa eletrônico; o sistema efetua o recebimento de uma conta e, no final, pergunta ao cliente se ele deseja pagar outra conta. Como os comandos utilizados para receber cada conta são os mesmos, eles ficam dentro de uma estrutura de repetição. E o programa pode ser utilizado tanto pelos clientes que desejam pagar apenas uma conta, quanto por aqueles que necessitam pagar várias contas.

INTERRUPÇÃO NOS LAÇOS: BREAK E CONTINUE

As linguagens de programação dispõem de dois comandos especiais para serem utilizados nas estruturas de repetição. São eles: `break` e `continue`. O `break` sai do laço de repetição, enquanto o `continue` retorna ao início do laço.

Estes comandos nos auxiliam no controle de execução dos comandos do loop. Observe a partir da ilustração da Figura 5.7, o que ocorre no laço de repetição no momento em que os comandos `continue` e `break` são executados.

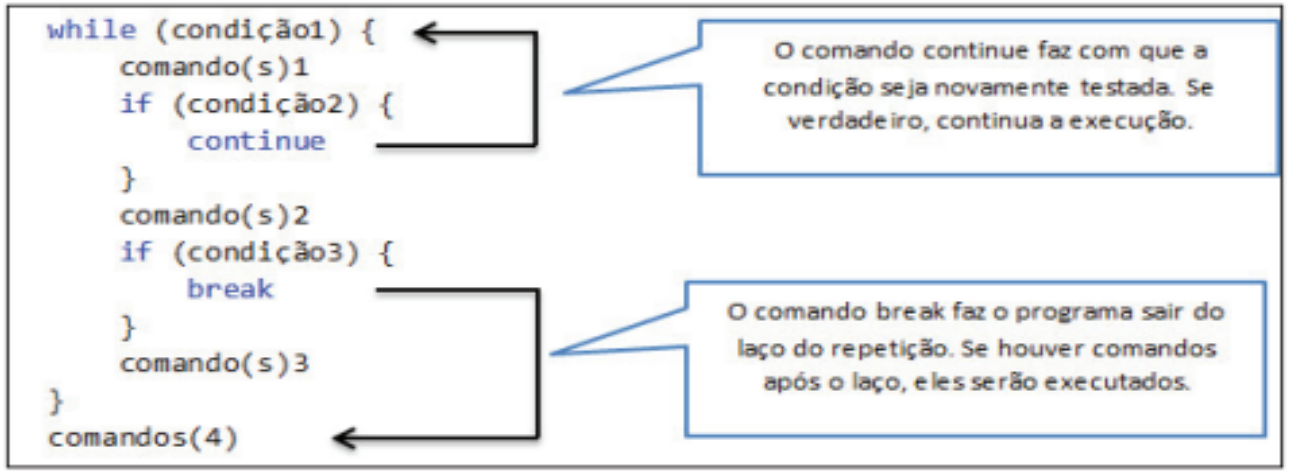

Figura 5.7 – Comandos continue e break modificam o fluxo dos comandos da repetição.

Os comandos `break` e `continue` podem ser utilizados nas três estruturas de repetição

CONTADORES E ACUMULADORES

O uso de contadores e acumuladores em um programa permite a exibição de contagens e totalizações. Essas operações são realizadas sobre os dados manipulados pelo programa. Os contadores ou acumuladores possuem duas características principais:

- A variável contadora ou acumuladora deve receber uma atribuição inicial (geralmente zero).
- A variável contadora ou acumuladora deve receber ela mesma mais algum valor.

A diferença entre os contadores e os acumuladores é que o contador recebe ele mesmo mais 1 (ou algum valor constante), enquanto o acumulador recebe ele mesmo mais uma variável.